Pump up the Volume: Practical Database Reconstruction from Volume Leakage on Range Queries

Paul Grubbs
pag225@cornell.edu
Cornell University

Marie-Sarah Lacharité
marie-sarah.lacharite.2015@rhul.ac.uk
Royal Holloway, University of London

Brice Minaud
brice.minaud@rhul.ac.uk
Royal Holloway, University of London

Kenneth G. Paterson
kenny.paterson@rhul.ac.uk
Royal Holloway, University of London

ABSTRACT

We present attacks that use only the volume of responses to range queries to reconstruct databases. Our focus is on practical attacks that work for large-scale databases with many values and records, without requiring assumptions on the data or query distributions. Our work improves on the previous state-of-the-art due to Kellaris *et al.* (CCS 2016) in all of these dimensions.

Our main attack targets reconstruction of database counts and involves a novel graph-theoretic approach. It generally succeeds when R, the number of records, exceeds $N^2/2$, where N is the number of possible values in the database. For a uniform query distribution, we show that it requires volume leakage from only $O(N^2 \log N)$ queries (cf. $O(N^4 \log N)$ in prior work).

We present two ancillary attacks. The first identifies the value of a new item added to a database using the volume leakage from fresh queries, in the setting where the adversary knows or has previously recovered the database counts. The second shows how to efficiently recover the ranges involved in queries in an online fashion, given an auxiliary distribution describing the database.

Our attacks are all backed with mathematical analyses and extensive simulations using real data.

CCS CONCEPTS

• **Security and privacy → Cryptanalysis and other attacks**; **Management and querying of encrypted data**;

ACM Reference Format:
Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth G. Paterson. 2018. Pump up the Volume: Practical Database Reconstruction from Volume Leakage on Range Queries. In *2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18), October 15–19, 2018, Toronto, ON, Canada.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3243734.3243864

CCS '18, October 15–19, 2018, Toronto, ON, Canada

ACM ISBN 978-1-4503-5693-0/18/10…$15.00
https://doi.org/10.1145/3243734.3243864

1 INTRODUCTION

Setting. In a recent ground-breaking paper, Kellaris *et al.* [14] initiated the systematic study of volume attacks against databases. Here, the setting is an adversary who is able to learn how many records are returned in response to queries made to a database. From just this information, the adversary tries to reconstruct the *database counts*, that is the exact number of records in the database having each particular value. As a secondary target, the adversary may try to learn the content of individual queries.

We stress that, in the envisaged setting, the adversary does not know the individual queries (these may be encrypted) and does not learn *which* records are returned in response to each query (so it does not have what is known as *access pattern leakage*). In some settings, the attacker may know something about the distribution on queries (for example, that they are uniformly distributed range queries), and it may also have access to some kind of reference distribution which represents (possibly inaccurate) side information about the distribution from which the database is drawn. However, we are also interested in attacks in the "bare" setting where the attacker has no ancillary information.

Such volume attacks may be quite easy to mount in practice. For example, the attacker might be located on the network between a client making queries and a server hosting the database, with all interactions between client and server being encrypted. Yet the communication pattern and volume of data on the wire from server to client may indicate how many records are returned from each query, since typical secure communications protocols like TLS do not attempt to hide the directionality or amount of data being transmitted (indeed, modern TLS cipher suites like those based on AES-GCM directly leak plaintext lengths in ciphertexts). This makes volume attacks possible. This setting is of course related to problems in the field of traffic analysis, such as mounting and preventing website fingerprinting attacks [7, 22]. Relatedly, the time taken to process database queries may act as a side channel to reveal the volume of responses.

As a second example of the relevance of volume attacks, the database server itself might be adversarial, with the client using advanced encryption techniques (possibly in combination with trusted hardware like SGX) to hide queries and access patterns from the server. Existing techniques for this purpose [4, 5] do hide the queries but tend to leak the access pattern, rendering them vulnerable to quite devastating attacks, see [14, 16]. The next natural evolutionary step in this area, then, will be to combine the

existing techniques with oblivious memory access techniques such as ORAM [9, 23] to hide the access patterns. However, not only would this degrade performance, but also such an approach might be of dubious security value, since volume attacks mounted by the server would still be possible and might have significant security impact. Moreover, as recent work has shown [10], SQL databases store the cardinalities of responses to past queries, so volumes may leak even to the so-called snapshot adversary who is only able to compromise the server for a short period of time and grab a copy of its memory. This is a weaker adversarial setting than the persistent attacker setting implicit to assuming an adversarial server.

The impact of volume attacks, if possible, can be serious. Although they cannot reconstruct the exact connection between individual records and their values, they do enable the *exact database counts* to be reconstructed, and this may represent significant leakage. As illustrative examples, consider a company's salary database leaking to a competitor, or a hospital's mortality data becoming exposed. This is even so when the adversary already has an approximation to the database distribution, since knowing exact counts can represent a much more serious privacy violation than merely having approximations to those counts. For example, by mounting the attack and recovering exact counts at different points in time, the adversary may be able to deduce the value of specific records of interest that were added or removed from the database. Furthermore, the exact database counts can leak important information about the values of specific outliers, which can then be de-anonymising. Indeed, the privacy risks of releasing precise database counts were among the core motivations of modern differential privacy research.

Range queries. In this work, we focus on database reconstruction using the volume leakage of *range* queries. Range queries are perhaps the simplest type of query beyond point queries, and constitute a central primitive in modern databases: for example, four queries in the TPC-H query benchmark (designed to reflect real workloads) contain explicit range queries. In the setting of range queries, data takes on numerical values in some range $[1, N]$ (the value of the left endpoint is fixed at 1 only for ease of exposition, and without loss of generality [16]). All range queries are of the form $[x, y]$ for some $1 \leq x \leq y \leq N$. When a range query $[x, y]$ is issued, all records with values $z \in [x, y]$ are returned in response to the query. Because we target *volume* attacks, recall that the adversary only sees the *number* of records returned by the query, and not the record identifiers, or the values x, y.

Existing work. Kellaris *et al.* [14] (KKNO) made an excellent first step in understanding volume attacks arising from range queries, formalising this style of attack and introducing a pair of algorithms that are each capable of performing database reconstruction. KKNO's attacks were the first to demonstrate that reconstructing database counts solely from the volume leakage of range queries is *possible*. However their attacks are severely limited in practice, for two reasons. First, KKNO's algorithm strongly relies on the assumption that range queries are drawn *independently* and *uniformly* at random. Real-world queries are not expected to be uniform, or independent. If these conditions are not met, KKNO's algorithm fails. In fact the uniformity assumption seems inherent to the KKNO algorithm, which exploits specific properties of that distribution; it is unclear how the algorithm could be adapted to more general distributions. A second limitation of KKNO's attack in practice is that it requires observing the volume leakage from $O(N^4 \log N)$ queries. For $N = 100$ for instance, this number represents about half a *billion* queries. For further discussion, see Appendix A.

It may seem at first that this query complexity is unavoidable. Indeed as shown in [14], any generic algorithm succesfully achieving database reconstruction from volume leakage must require as many as $\Omega(N^4)$ range queries. However, the example databases demonstrating this are certainly pathological, and one might wonder whether this is the true barrier to performance for *typical* databases in which the values in records are drawn from some reasonable distribution. Our work shows that it is not.

Our results. Because of the ease with which they can be mounted, their real-world impact on privacy, and their likely future importance, it is vital to understand volume attacks better. This is what we set out to do in this work. Our focus is on making database reconstruction (DR) attacks using volume leakage from range queries more practical.

In this direction, our main result is a volume-based DR attack for range queries that does not rely on any uniformity or independence assumptions on the query distribution. Instead, it only needs to observe each distinct volume at least once, regardless of how queries are drawn. The former property makes our attack much more practical than those of KKNO, which as discussed above crucially rely on a uniformity assumption. The latter property leads to a substantial reduction in the number of queries needed for a DR attack, since now only the "coupon collector number" of queries needs to be seen for the query distribution. For example, if for the purpose of comparison to KKNO's algorithm, we assume that queries are uniform, then our algorithm only requires $O(N^2 \log N)$ queries, instead of $O(N^4 \log N)$ for KKNO's.

In more detail, our approach reduces the problem of DR to finding a clique in a certain graph that is constructed from the volume leakage. By applying suitable preprocessing, in practice, we find that we actually often end up in the situation where clique finding in the graph is trivial, avoiding the need for expensive clique-finding algorithms. We evaluate the performance of our algorithm using real medical datasets obtained from the US government Healthcare Cost and Utilization Project (HCUP) Nationwide Inpatient Sample (NIS).

Our attack has two main limitations. First, as noted above, we assume that every range query must be issued at least once. This is certainly a strong assumption. Nevertheless, it is considerably weaker than the assumption required by KKNO's attack, which needs that every query should be observed multiple times (roughly N^2 times on average) so that the *exact* frequency of every volume can be determined. Furthermore, we believe it is reasonable to expect that a secure encrypted database should remain secure even when every range query has been issued.

The second main limitation of our attack is that it does not succeed for *all* databases. Indeed that would be impossible, since it would then have to require $\Omega(N^4)$ queries due to the lower bound from [14], as discussed earlier. Instead we aim to cover typical parameter regimes that include many real-world databases. Assumptions on the database required by our algorithm are twofold. First, if we wish to recover the exact count of every value in the

database, then we must assume that the database is *dense*, in the sense that every value must be taken by at least one record (equivalently, there is no value with a zero count). However, our attack does extend to the non-dense (or *sparse*) case, with the limitation that it only recovers non-zero database counts. This point is discussed in more detail in Section 3.3.

Second, our attack does not succeed for all parameter regimes, although our experiments show a high success rate on real-world datasets. In addition to experiments, to help provide insight about parameter regimes where our attack succeeds, we build a statistical model of how the number of records R and the number of values N influence the adversary's view of volumes, and how this affects the success of our attack. For example, if for the purpose of the model we assume a uniform distribution of values across records, and a uniform distribution on ranges, then our model predicts that the number of records R required for our attack to succeed should be $\Omega(N^2)$ when the leakage from $O(N^2 \log N)$ queries is available. This estimate matches with what we observe in practice in our experiments, despite several idealisations made in building the model: we find that when R exceeds about $N^2/2$, our clique-finding algorithm works extremely well in practice (given volume leakage from enough queries), but its performance declines markedly when R becomes significantly lower than $N^2/2$.

In addition to our main attack, we propose two ancillary attacks. The first of these ancillary attacks considers a setting where the adversary has already recovered exact database counts, for example as a result of running our main attack. Then we assume the database is updated with a new record. We propose an algorithm to deduce the value of that new record, purely from observing the volume of range queries, as in the previous attack. This enables the adversary to update its knowledge of the database on the fly as new records are added.

Like our main attack, the algorithm we propose does not require an assumption on the query distribution. However if for the sake of analysing the performance of the algorithm, we do make the assumption that queries are uniformly distributed, then we are able to show that only $O(N)$ queries are needed for our algorithm to recover the value of the newly added recorded, provided again that R is $\Omega(N^2)$. We stress that the uniformity assumption is needed only for analysis; the algorithm still works well without it. We go on to show that even better performance can be achieved if only approximate recovery of the new value is desired. This analysis again supports our experimental results using HCUP datasets. For example, our experiments show that on a real-world hospital database of about 20000 records, the median number of queries needed to ascertain the age of a patient in a newly added record to within 10 years is only 17 queries; after 57 queries it is known within just 2 years.

Our second ancillary attack shows how to efficiently recover the ranges involved in queries in an online fashion, given a reference distribution for the database. This reference distribution could be obtained by a successful DR attack, but it could also be an inaccurate estimate obtained from a related dataset or a previous breach. The attack relies on the following idea: given the volume leakage for a query, we can compare that leakage to volumes obtained synthetically from all the ranges in the reference distribution. In our attack, we just output the set of all ranges whose volumes are close (in a well-defined sense) to the leaked volume. Although simple, this idea turns out to be powerful. It is also amenable to analysis. For example, assuming the database is drawn exactly from the reference distribution, we are able to prove that the output set always contains the correct range, except with some small (and tunable) error. This follows from an application of the Dvoretzky-Kiefer-Wolfowitz inequality, a Chernoff-type bound on the maximum distance between the empirical and true CDFs of a distribution. Surprisingly, our simple "CDF matching" algorithm continues to work well even when the reference distribution is not particularly accurate. To demonstrate this, we again use the HCUP datasets; we compile a reference distribution by aggregating data from one year, and use it in attacks against individual hospitals from other years. For more than 80% of hospitals our attack is able to correctly eliminate all but fifteen possibilities for some queries on the AGE attribute (which has 4186 possible queries).

In its entirety, our work shows that volume attacks, perhaps not yet considered a serious security threat because of unrealistic assumptions or poor performance, should be considered a real concern in practice, not only in advanced settings (like an honest-but-curious database server) but even in basic settings such as a network-based or snapshot adversary. Our work should also serve as a warning for researchers developing new database encryption schemes: simply hiding access patterns is not enough; volumes must be hidden too.

2 BACKGROUND AND SETTING

The setting we consider has two parties: a *client* and a *server*. The client stores no information locally and the server stores a database that the client queries. In this paper we will treat the database as a sequence of values between 1 and N (hereafter, "records"), but will not assume anything about how records are stored or accessed. We assume client-server communication is unbreakable and that the queries and responses sent between the client and server reveal no information except for the number of records in the response. This is an extremely conservative setting: the attacker neither knows nor can issue any queries.

Reconstruction attacks. Call the number of records with a given value the *count* of the value. As discussed in the introduction, our main attack targets *database reconstruction (DR)*, which is to say it attempts to recover the counts of all values. KKNO observed that counts can only be recovered up to *reflection*, meaning that for any value k, the recovered count could be for k or $N + 1 - k$. When no assumptions can be made about the number of records or the counts of individual elements (we will call this *sparse*), Kellaris *et al.* proved that $\Omega(N^4)$ queries are *required* to perform DR generically. We will also study two other attack types. The first, *update recovery*, learns the value of a single record added after the database is reconstructed. The second is *query reconstruction*, which tries to reconstruct the queries rather than the database.

Notation and terminology. Recall that N is the number of possible values. We assume (without loss of generality) that set of possible values is $[1, N]$. The number of records is denoted by R. We let $[x, y]$ for $1 \le x \le y \le N$ represent a query for all records whose value is in the closed interval from x to y. There are $N(N + 1)/2$

possible range queries. The *volume* of a range query $[x, y]$ is the number of records whose value lies in $[x, y]$. We also say that these records *match* the range query. We call *volumes* the set of integers that are the volume of some range. We denote the volume of a range $q = [x, y]$ by $\text{vol}(q)$. We say that a database is *dense* iff every value is taken by at least one record. We will use standard asymptotic notation (O, Ω, etc.) as well as "tilde" asymptotic notation like $\tilde{O}$, which simply hides polylog factors. log() denotes the natural logarithm.

Assumptions. We assume that the total number of records, R, is known by the adversary. We believe this is a reasonable and conservative assumption. Releasing the aggregate size of a database (even one containing sensitive information) has little or no privacy implications in most settings, and the value R may in fact be publicly available. Even if the information is not public, an adversary can infer R using the on-disk size of a database or by observing network traffic while a database is restored from backup.

We also assume the adversary knows the total number of possible data values N in the field targeted by the range queries. Note that this number does not depend on the database under attack, but only on the type of data being targeted. Other assumptions required by our main attack are discussed in Section 1.

Unless otherwise specified, we will never assume an attacker knows either the query distribution or the database distribution. An assumption on the query distribution becomes necessary when it comes to analysing the query complexity of the attacks though: indeed the client could otherwise repeat the same query forever, and the adversary would never learn anything new. To give meaningful and clear analyses we will thus sometimes assume a uniform distribution; however, in every case the algorithm itself does not require that assumption to succeed. We make these assumptions to provide analytical insight into a "typical" behavior of the algorithm. Further, our attacks are evaluated on real-world non-uniform datasets.

3 PRACTICAL VOLUME-ONLY RECONSTRUCTION ATTACKS

In this section, we describe and analyze our main result, namely a practical database reconstruction attack using only the volume leakage of range queries. Our attack uses only a *set* of range volumes as input, and does not use any distribution-dependent frequency information. In particular, the success of the attack is not dependent on knowing the query distribution.

We begin this section by discussing the requirement of data density, presenting the key idea behind our algorithm – identifying elementary ranges – and then providing an overview of the algorithm's steps. We explain it in detail in Section 3.1. We then analyze it in Section 3.2 and present the results of practical experiments in Section 3.3.

Data density. The main setting of our algorithm is the case where the database is dense; that is, where every value in $[1, N]$ matches at least one record. Our algorithm succeeds on dense databases, and can also succeed if the database is not dense. In that case, it is not possible to recover the counts of all values, because it is impossible to learn which values are matched by zero records from just a set of volumes of range queries. Therefore, we define success as recovery of the *non-zero* counts of all values in the correct order; that is, the only missing information is precisely the set of values with zero matching records. This still reveals a considerable amount of information to the adversary. Some knowledge of the database distribution may enable reconstruction of all counts. This is discussed further in Section 3.3. Nevertheless, the main focus of our attack is dense databases, where recovering the counts of all values is possible with just a set of range query volumes.

Elementary ranges. Consider the ranges $[1, 1]$, $[1, 2]$, . . ., $[1, N]$ – let us call them *elementary ranges*. Knowing the volumes of these N ranges is necessary and sufficient for reconstruction: if we know the volumes of $[1, 1]$, $[1, 2]$, . . ., $[1, N]$, then the number of records that have value k is the difference between the $(k-1)$-st and k-th element in the list (treating the 0-th element as zero). The goal of our algorithm will therefore be to identify the volumes of these elementary ranges among the set of all volumes.

Our approach stems from the following observation: every range $[x, y]$ can be expressed either as $[1, y]$ (if $x = 1$) or as $[1, y] \setminus [1, x-1]$ (if $x > 1$). In other words, every range is either an elementary range, or can be expressed as the set difference of two elementary ranges. From the point of view of volumes, this means that every volume is either the volume of an elementary range (an *elementary volume*), or the difference of the volumes of two elementary ranges. Conversely, the set difference of two elementary ranges is a range, and so the difference (in absolute value) of the volumes of two elementary ranges is itself an observed volume. Elementary ranges are also *R-complemented*. That is, if v is the volume of an elementary range, then $R-v$ must also be the volume of a range. This holds because the complement of an elementary range is also a range. Note that the range $[1, N]$ of maximum volume R may not be R-complemented; however by convention and to avoid special cases later on, we shall say that it is R-complemented.

In summary, the volumes of elementary ranges have the following strong properties among the set of all volumes:

(1) Every volume occurs as the volume of an elementary range, or the difference of the volumes of two elementary ranges.
(2) Conversely, the difference (in absolute value) of the volumes of any two elementary ranges is a volume.
(3) If $v < R$ is the volume of an elementary range, then $R - v$ is also a volume.

In fact, if we include 0 as the volume of an elementary range (whether 0 was an observed volume or not), these properties imply that the set of pairwise differences of the volumes of elementary ranges is exactly the set of *all* volumes.

Building a graph. In order to exploit the previous three properties of elementary ranges, we build a graph as follows. The nodes (a.k.a. vertices) of the graph are the R-complemented volumes. Two nodes are connected by an edge iff their absolute difference is also a volume; we label the edge by that difference. Property (3) implies that elementary volumes must appear as nodes in the graph. Property (1) implies that every volume must occur as a node or an edge in the subgraph induced by the elementary volumes. Property (2) means that every pair of elementary volumes must share an edge: that is, the elementary volumes form a clique in the graph.

Algorithm overview. One natural approach is to build the graph and run a clique-finding algorithm to identify elementary volumes.

Empirically, this approach suffices in some cases but has three drawbacks. First, in many real-world datasets the edge density of the graph is high and clique-finding does not terminate in a reasonable time frame (recall that finding a maximum clique is NP-complete). Second, in some cases the solution clique is not maximal, and clique-finding would not suffice. Third, this approach does not use Property (1). For these reasons, our algorithm uses generic clique-finding only as a last resort.

Instead, we run a *graph pre-processing* phase, which exploits both properties 1 and 2 to simultaneously identify nodes that *must* be elementary volumes, named *necessary* nodes; and prune nodes that *cannot* be elementary volumes, to progressively reduce a set of *candidate* nodes. This step is iterated until the sets of necessary and candidate nodes stabilize. Our experiments show that in many cases, this pre-processing step suffices to identify the set of elementary volumes. However, in the cases that it does not suffice, we then run a *clique-finding* step that seeks to extend the set of necessary nodes into a clique within the subset of candidate nodes while satisfying properties 1 and 2.

We now explain the steps of our algorithm in detail.

3.1 Reconstruction Algorithm

The idea of our algorithm is to use the properties of elementary ranges to identify their volumes among the set of all volumes. As explained earlier, this information is then enough to reconstruct database counts up to reflection (cf. Section 2). The main constraint of our algorithm is that it requires the set of all volumes. In the context of an adversary observing volume leakage, this means that (in general) every range query must have been issued at least once.

Step 1: Obtaining the set of query volumes. The first step of the algorithm is to collect all volumes. Let V be the set of all observed volumes. We can bound the number of possible distinct volumes. If the data is dense (i.e., each of the N values occurs at least once), then $N \leq |V| \leq N(N+1)/2$. If the data is sparse (iff the volume 0 appears in V), then the number of values that appear in the database (with non-zero counts) is at least $N_{\min} \stackrel{\text{def}}{=} -0.5 + 0.5 \cdot \sqrt{1 + 8 \cdot (|V| - 1)}$.

The query complexity of this step depends on the query distribution. No constraint on the query distribution is required, other than the fact that every query should have non-zero probability. The adversary does not need to know the query distribution. In principle, the adversary could even try to run the attack and see whether it succeeds to determine whether all volumes have been observed yet. If a query distribution is assumed, one can give bounds on the number of queries necessary before all volumes have been collected with high probability. This is discussed further in Section 3.2.

Step 2: Graph pre-processing. Given the set V of all possible query volumes, we form an initial set of candidate elementary volumes, V_{cand}. First, if the data was not dense (iff 0 appeared in V), then we remove it entirely from V for simplicity. Let V_{cand} be the set containing R and all volumes that have an R-complement:

$$V_{\text{cand}} \stackrel{\text{def}}{=} \{R\} \cup \{v \in V : R - v \in V\} \setminus \{0\}.$$

This set will contain R, pairs of volumes, and maybe the singleton volume $R/2$ if R is even and this volume was observed. This is the initial set of nodes. It must contain the volumes of the elementary ranges $[1,1]$ through $[1,N-1]$ because of their complementary ranges $[2,N]$ through $[N,N]$. It must also contain the elementary volume R for range $[1,N]$.

We place an edge between two node candidates iff their absolute difference is an observed volume: the set of edges E is defined as

$$E \stackrel{\text{def}}{=} \{(v,v') \in V_{\text{cand}} \times V_{\text{cand}} : |v - v'| \in V\}.$$

Form the graph $G = (V_{\text{cand}}, E)$ with node set V_{cand} and edge set E. In the full version of this paper, we present an analytical model to estimate the number of vertices and edges in the graph.

In this pre-processing step, we prune the set of nodes V_{cand} and identify a set of nodes $V_{\text{nec}} \subseteq V_{\text{cand}}$ that must be in the clique. We present an example of graph-preprocessing in Figure 1. Subfigure (a) shows the initial graph – for the moment, ignore its distinguished nodes.

Let $v_{\min}$ be the smallest R-complemented volume. It must be an elementary volume, up to reflection. Indeed the largest volume strictly smaller than R can only be the volume of $[1,N-1]$ or $[2,N]$, since every other range strictly within $[1,N]$ is included in one of those two ranges. It follows that the smallest R-complemented volume is either the volume of $[1,1]$ or the volume of $[N,N]$. Since we can only reconstruct the database up to reflection (cf. Section 2), we break the reflection symmetry by assuming that it is the volume of $[1,1]$ – which is correct up to reflection. In this respect, note that by the reflection symmetry, in addition to the N-clique induced by the volumes of queries $[1,1],[1,2],\ldots,[1,N]$, the graph will contain another N-clique generated by the volumes of the queries $[N,N],[N-1,N],\ldots,[1,N]$; reconstruction up to reflection is equivalent to recovering one of these two solutions.

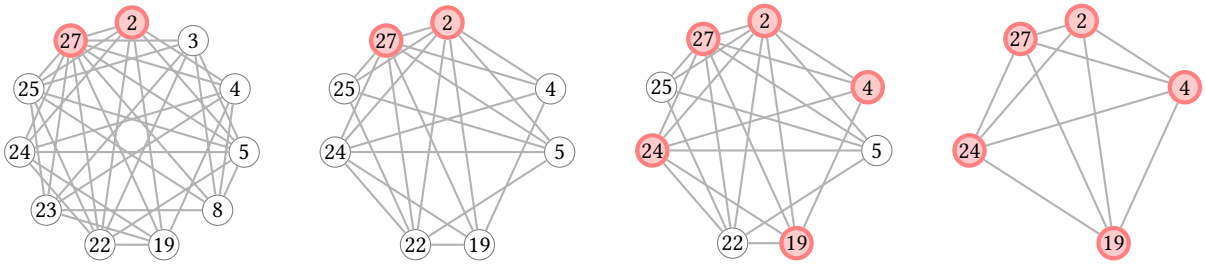
Therefore, we initialize the set of necessary nodes V_{nec} to contain $v_{\min}$ and R. These two nodes are highlighted in subfigure (a) in the example in Figure 1.

Next, we repeatedly perform the two following steps until they do not yield changes in the sets V_{cand} and V_{nec}: (1) eliminate node candidates that are not adjacent to all necessary nodes, and (2) identify necessary nodes based on volumes that arise only as one node candidate or edges incident to it.

Below, we briefly describe these two steps. For details of graph pre-processing, see Algorithm 1 in Appendix B.

Eliminating candidate nodes: If any node in $V_{\text{cand}} \setminus V_{\text{nec}}$ is not adjacent to all nodes in V_{nec}, then it cannot be an elementary volume, so remove it from V_{cand}. In the example in Figure 1, we see in subfigure (b) that three nodes have been removed in this way.

Identifying necessary nodes: There are three ways to extend the set of necessary nodes. First, if the set of node candidates is as small as it can be ($N_{\min}$ if the data is sparse, or N otherwise), then all candidate nodes must be necessary ($V_{\text{cand}} = V_{\text{nec}}$). Second, if any non-complemented volume arises only as edges incident to a single non-necessary node candidate, then this node must correspond to an elementary volume and is therefore added to V_{nec}. In the example in Figure 1, we see in subfigure (c) that nodes 4 and 19 have been added to the set of necessary nodes because non-complemented volume 15 arises only as an edge between them. Finally, if any non-necessary node candidate arises only as itself or as edges incident to itself, then it must correspond to an elementary volume and is added to V_{nec}. We see in subfigure (c) that node 24 was added to



(a) The largest observed volume is R=27, so we initialize V_{cand} to the set of R-complemented volumes and V_{nec} to $\{v_{\min}, R\}$=$\{2, 27\}$.

(b) Eliminate candidate volumes 3, 8, and 23 since they are not adjacent to both nodes in V_{nec}.

(c) 4 and 19 are necessary since 15 arises only as their difference. 24 is necessary since it does not arise as a difference of candidate volumes, only as a candidate volume itself.

(d) The number of necessary volumes is N, so pre-processing succeeded. These elementary volumes correspond to element counts 2, 2, 15, 5, 3 (or 3, 5, 15, 2, 2).

Figure 1: An example of pre-processing for a database with $N = 5$ distinct elements having counts 3, 5, 15, 2, and 2. The set of all possible range query volumes is $\{2, 3, 4, 5, 8, 15, 17, 19, 20, 22, 23, 24, 25, 27\}$. Nodes corresponding to necessary elementary volumes have thicker borders and red shading.

the set of necessary nodes for this reason. The example finishes in subfigure (d) when all remaining non-necessary candidate nodes are removed since they are not adjacent to all of the necessary nodes.

In Appendix B we prove a straightforward lemma that shows this procedure is correct: it does not eliminate any elementary volumes from the set of node candidates, and all necessary nodes correspond to elementary volumes.

Step 3: Clique-finding. At this point, we have two sets of volumes, V_{nec} and V_{cand}, satisfying $V_{\text{nec}} \subseteq V_{\text{elem}} \subseteq V_{\text{cand}}$, and we know a lower bound, $N_{\min}$, on the size of the clique formed by V_{elem}. As we will see when we present our experimental results, when the data is dense, the pre-processing in Step 2 often found a clique that generated all volumes in V (i.e., the sets it found satisfied $V_{\text{nec}} = V_{\text{cand}}$). This is the case in the example of Figure 1. When that is not the case, however, we must find a clique of size at least $N_{\min}$ in the graph induced by V_{cand} that generates exactly all volumes in V. There may be multiple such cliques. Although the clique of the elementary volumes V_{elem} must be a subclique of a *maximal* clique, it is not necessarily a subclique of a *maximum* clique (the largest maximal clique).

Our approach is motivated by the following observation: since the clique we want to find must include the nodes in V_{nec}, which already form a clique, we can reduce our problem to finding the rest of the clique in the subgraph of non-necessary candidate nodes – that is, the subgraph induced by $V_{\text{cand}} \setminus V_{\text{nec}}$. This second part of the clique must have the following properties. First, it must generate all *missing* volumes – the volumes in V that do not arise as nodes or edges in the subgraph induced by V_{nec} – and no volumes outside of V. The missing volumes could arise either as edges between the nodes of this clique part, or as edges between its nodes and the nodes in V_{nec}. Second, if the number of elementary volumes is between $N_{\min}$ and $N_{\max}$, then this clique part must have size at least $\max\{0, N_{\min} - |V_{\text{nec}}|\}$ and at most $N_{\max} - |V_{\text{nec}}|$. Given such a clique in the subgraph of non-necessary candidate nodes, we recover the elementary volumes by combining it with V_{nec}.

Pseudocode is given in Algorithm 2 in Appendix B. Algorithm 2 returns a set of lists of volumes that (1) include $v_{\min}$, (2) have size between $N_{\min}$ and $N_{\max}$, (3) generate exactly the volumes in V and no others, and (4) are not supersets of any other list of volumes in the returned solution. This set of lists of volumes must include V_{elem}, or a subset of V_{elem} if the data was sparse. See Lemma B.2 for the full statement of correctness.

3.2 Analysis of the Algorithm

Time complexity. The pre-processing step increases V_{nec} or decreases V_{cand} at each step; since there are at most $N(N+1)/2$ volumes, it follows that this step iterates $O(N^2)$ times. The bulk of the time complexity comes if clique-finding is run.

Finding maximal cliques. In general, a graph on n nodes can have an exponential number of maximal cliques [19] – this clearly seems incompatible with our goal of *practical* reconstruction attacks. When the number of nodes is small, however, it is still feasible to enumerate all of the maximal cliques with an algorithm such as Bron–Kerbosch [2]. For larger domains, there exist logarithmic time algorithms to sample *one* maximal clique at a time [17].

Finding minimal subcliques. In the worst case, if the data is not dense, the check on line 40 in Min_Subcliques in Algorithm 2 will be carried out for $2^{|V_k|} - 1$ subcliques. In Section 3.3, we evaluate a variant that either returns all subcliques, not just minimal subcliques, or fails if it is impractical to do so.

Query complexity. We must assume something about the distribution to analyze the query complexity of collecting all volumes. In the case of a uniform query distribution, this is the classic coupon collector's problem; because there are $O(N^2)$ possible queries, coupon collection implies that $O(N^2 \log N)$ queries suffice (and the constants are small).

For a non-uniform distribution, if the least likely range has probability $\frac{\alpha}{N(N+1)/2}$, then a straightforward adaptation of coupon collection analysis shows that $O(\alpha^{-1}N^2 \log N)$ queries suffice. Computing coupon collector bounds for arbitrary distributions is straightforward, but somewhat tedious: Flajolet *et al.* [8] give a generating function for it. They also give a closed-form solution for one distribution of practical importance, namely the standard Zipf [25] distribution (where the k-th most likely element has probability proportional to $1/k$). Somewhat surprisingly, their results imply that even if the query distribution is Zipf the query complexity of collecting all volumes is only $O(N^2 \log^2 N)$. This means that even with a very skewed distribution the query complexity of our attack is not much higher than with a uniform distribution.

Analytical model. In the full version of this paper, we build an analytical model of the graph underpinning the algorithm to provide insight into its behavior. We assume the records are sampled i.i.d. from a fixed distribution. The database counts then follow a multinomial distribution, which can be modelled (with only a factor of 2 loss in some cases [18]) by a series of independent Poisson variables. Because a sum of Poisson variables is itself a Poisson variable, the number of records matching any given range is also a Poisson variable. Using properties of the difference of Poisson variables, we can then approximate the collision probability between volumes, and ultimately compute estimates of the number of distinct volumes, nodes, and edges in the graph.

To compute concrete bounds we assume the database distribution is uniform. We show that the number of volume collisions can be approximated by $N^3/(4\sqrt{\pi R})$ and our experiments support this estimate. In particular, having no volume collision whatsoever would require $R = \Omega(N^6)$, which is only reasonable for very low values of N. This shows the importance of using algorithms that are resilient to the fact that volumes do collide, and hence do not in general identify a unique range; as is the case of our algorithm.

We also show that the ratio R/N^2 relates both to the ratio of collisions among volumes, and to the edge density of the graph, suggesting that it is a critical quantity for assessing the success of our algorithm. Our experiments show that when this ratio is 1/2 or more, our algorithm typically succeeds easily (even on non-uniform age data), while when it is much lower than 1/2 it typically fails.

Furthermore if we model the graph as a random graph, as far as its clique number is concerned (and disregarding the existence of the two N-cliques stemming from elementary ranges), then we show that for the two solution cliques stemming from elementary volumes to be of maximum size among all cliques (and hence uniquely identifiable as such), it should be the case that $R = \Omega(N^2)$. We refer the reader to the full version of this paper for more details.

3.3 Experimental Evaluation

In this section, we present an experimental evaluation of the reconstruction attack from the previous section. We simulate an attacker who has observed enough queries to see all possible volumes of range queries. We implemented our algorithms in Python and used the `graph-tool` [21] and `NetworkX` [12] packages for finding cliques or maximal independent vertex sets.

Datasets and methodology. We test our algorithm on various attributes from three years of medical records from the US government's Healthcare Cost and Utilization Project (HCUP) Nationwide Inpatient Sample (NIS). The attributes we chose to extract have domain sizes that range from $N = 4$ to $N = 366$ and they are all attributes on which range queries are meaningful. For more information about these datasets and how we extracted attributes, see Appendix C. Each of the three years includes patient discharge records from about 1000 hospitals, giving us 3000 datasets for most attributes. (Some were not available in all years.)

We say the attack succeeds if there is a single solution output by Algorithm 2, and it is the *set* of elementary volumes (up to reflection). For dense datasets (where every value appears at least once and no range query has volume 0), this means that all element counts have been recovered exactly, up to reflection. For sparse datasets, this means that all non-zero element counts have been recovered in order (up to reflection), but it is not known which elements did not appear in the database – the attacker must make a decision about *which* values were not observed. In our evaluation of step 2, we discuss and evaluate one such strategy, which uses a small amount of auxiliary information, for assigning the recovered counts to a subset of elements in the domain in the sparse case.

Step 1 evaluation. The first step of the attack is to observe enough queries to see all possible range query volumes. The number of queries that this entails depends on the query distribution, as discussed in Section 3.2. For instance, if the query distribution is uniform, then the expected number of queries is $O(N^2 \log N)$.

Step 2 evaluation. For each dataset-attribute combination, we ran Algorithm 1 to obtain sets of necessary elementary volumes, V_{nec}, and candidate elementary volumes, V_{cand}. The plot in Figure 2 shows, for each attribute, the average[1] number of datasets for which pre-processing was sufficient for the attack to succeed. For all attributes except AGE and AGEDAY, pre-processing correctly identified the non-zero element counts in order (up to reflection) for the vast majority of datasets. The difference in patterns on the bars indicates which datasets were dense. Attributes with smaller domain sizes, e.g., AMONTH, MRISK, SEV, and ZIPINC, were dense most of the time. The attributes with the largest domain, LOS and AGEDAY, were dense in fewer than 0.01% of datasets.

Pre-processing recovered the set of elementary volumes for at least 90% of all dense datasets for each attribute except AGEDAY. This attribute had a single dense dataset in each 2004 and 2008 that required clique-finding.

For sparse datasets, recovering the set of all non-zero counts provides a lot of information. Combining it with some rudimentary information about the database distribution can lead to recovering *all* element counts, just like in the dense case. For instance, one might guess that the length of stay (LOS), the number of chronic conditions (NCHRONIC), and the number of procedures (NPR), might be 0 most frequently, then decrease. To illustrate just how valuable knowing the set of elementary volumes could be when combined with a tiny bit of knowledge about the domain, we evaluate the following strategy for assigning counts to elements: simply

[1]For attributes that were available in more than one year (as noted in Figure 7 in Appendix C), results were very similar, so we averaged the counts.

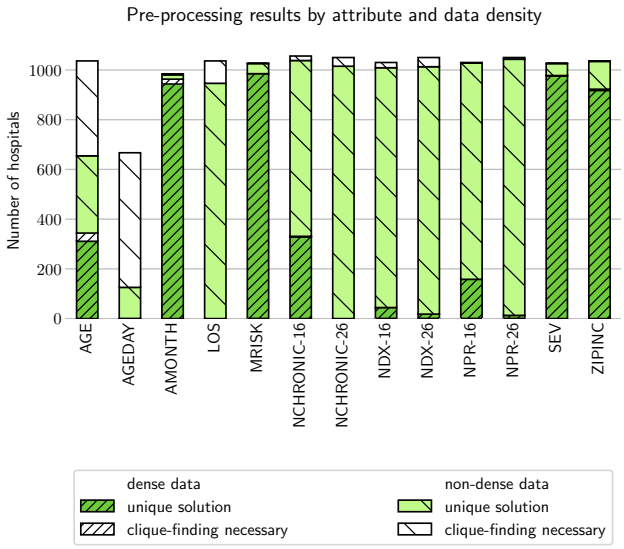


Figure 2: Pre-processing success and data density by attribute.

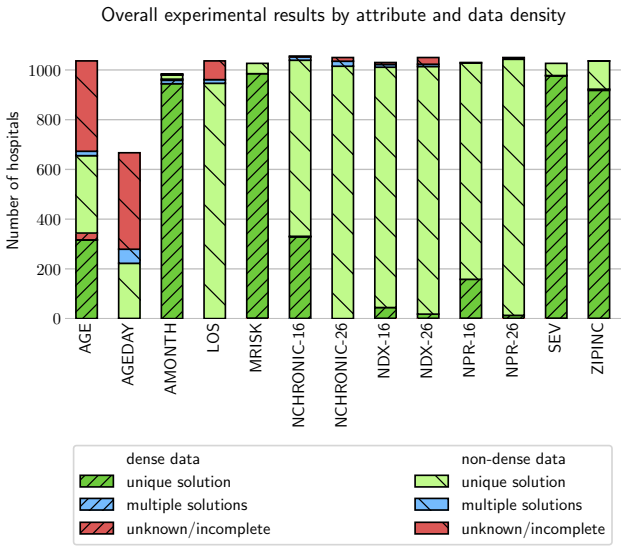


Figure 4: Overall results of the practical reconstruction attack.

guess that they correspond to the first values in the domain. The results are displayed in Figure 3. We juxtapose the success of our simple strategy for LOS, NCHRONIC, and NPR with its mediocre results for the number of diagnoses, NDX, which is 1 more frequently than 0, and thus our strategy is not suitable for it.

Step 3 evaluation. Lastly, we ran Algorithm 2 (with some modifications) on the few dataset-attribute combinations for which pre-processing did not find a unique solution. First, we modified Get_Elem_Volumes to return *all* solutions, not just minimal ones, by replacing Min_Subcliques on line 20 with All_Subcliques_P (described starting on line 15 in Alg. 3 in Appendix B). However, for the sake of a more practical attack, we allowed Get_Elem_Volumes to return an incomplete list of solutions, or to fail entirely.

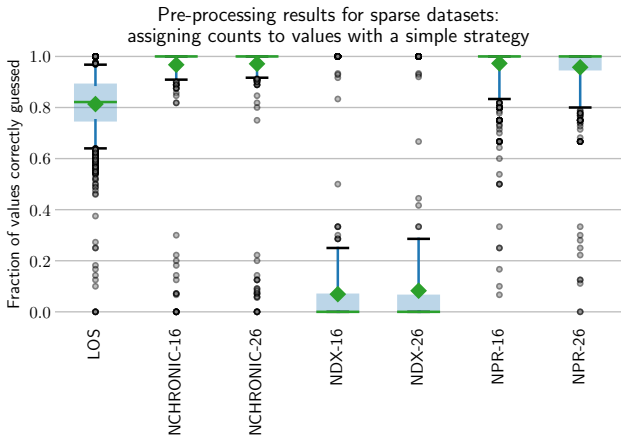


Figure 3: Extending pre-processing success for sparse datasets. The fraction of correct values is out of the actual number of values for each dataset.

Our probabilistic variant of Find_Maximal_Cliques is described starting on line 1 of Algorithm 3. Specifically, line 14 of Algorithm 2 is replaced with Find_Maximal_Cliques_P($\text{CAND}_{nn}, M_{\min}, V$). For graphs with 20 or fewer nodes, we used the `find_cliques` routine from the `NetworkX` Python module (line 4) [12]. For graphs with more nodes, we sampled maximal cliques one at a time, 1000 times, (line 7) using Luby's efficient parallel algorithm for maximal independent sets, implemented as the `max_independent_vertex_set` routine from the `graph-tool` Python module [21].

The three ways in which our variant may fail entirely are (i) Find_Maximal_Cliques_P fails to find any maximal cliques of size at least $M_{\min}$ (line 11), (ii) we found such cliques, but none of them generated the set of missing volumes (line 13), or (iii) there were such cliques that generated the set of missing volumes, but for all of them, it was impractical (line 17) to find all of their subclique solutions.

Figure 4 shows the overall attack results. Success, in green, occurs when pre-processing or clique-finding finds the solution and it is unique – there is a single clique whose size is in the right range that generates all observed volumes. Multiple cliques, in blue, arise when clique-finding has found all such solutions, but there is more than one, so that the correct solution cannot be precisely determined. Failure, in red, arises either when Algorithm 2 returns FAILURE or {} or when we sampled maximal cliques using Luby's algorithm (line 7) and may not have found all of them.

In our experiments, the most common reason for failure overall was (iii): it was impractical to find all subcliques (about 60% of failures or incomplete cases). The second most common overall reason for failure was (ii), not finding any cliques that generated all missing volumes (about 36%). However, as one might expect because of the bound on line 17 in All_Subcliques_P, the attributes with fewer possible values (e.g., AGE with $N = 91$ compared to AGEDAY with $N = 365$) failed more often due to no cliques generating all volumes as opposed to too-big cliques.

Conclusions. Overall, our experiments indicate that our clique-finding approach yields overwhelming success in reconstructing counts of dense datasets – and that in most cases, no expensive clique-finding is even required (see the white bars corresponding to dense data in Fig. 2). For sparse data, the success of this approach mainly depends on what auxiliary information is available to the attacker. We showed how an attacker can leverage rudimentary knowledge of a distribution (e.g., that the most frequent values are the smallest) to correctly assign exact counts to values (see Fig. 3).

4 UPDATE RECOVERY ATTACK

In this section, we consider an attack in the following setting. We assume that the adversary knows the database counts, via either the reconstruction attack from the previous section or a one-time compromise of the database. Now suppose that a new record is added into the database, and that the attacker learns this. The attacker could detect such an update for example because an update query may have a different volume than a range query; the attacker could also infer indirectly that an update has occured because he observes volumes that were not possible for the original database counts. In this context, we propose an attack to recover the value of the newly added record using only the volume leakage of range queries issued after the update.

Note that in order to fully recover the value of the new record, the attack assumes that enough range queries are issued by the client before any further update is made. Thus the attack as it stands will fail if updates are made in close succession. We leave the treatment of frequent or simultaneous updates for future work.

On the other hand, if there are enough range queries for our attack to fully recover the value of a new record after it is added, and before the next update, then database counts are fully known before the next update. It follows that the attack can be repeated for the next update. Thus if database updates are rare relative to range queries, then the attack allows an adversary to update its view of database counts on the fly as updates are made.

As a first idea to recover the value of the new record, one could re-run the database reconstruction attack and compare the original and new counts. This has unnecessarily high query complexity—our attack in this section is orders of magnitude more efficient analytically and experimentally.

Like our main attack, our update recovery algorithm does not require a uniform query distribution. If we do make that assumption for the purpose of analysis, then the algorithm is amenable to analysis in the same model as our main attack. Recall that in that model, our main attack required $O(N^2 \log N)$ queries for full reconstruction. In the same model, our update recovery algorithm only requires $O(N)$ queries to recover the value of the new record exactly. Furthermore the same algorithm is able to approximate the value of the new record quite quickly: our model predicts that the value of the new record can be approximated within an additive error ϵN, for any $\epsilon > 0$, after observing $O(1/\epsilon)$ queries; once again the observed behavior in our experiments on real-world data matches this prediction.

4.1 Update Recovery Algorithm

The idea of the attack is as follows. First, because the adversary knows all database counts for the original database, it knows the volume of every range query on that database. Now suppose that a new record is added, and the adversary then observes the volume of some range query.

Assume that the volume of that query is not equal to the volume of any range for the original database (i.e. before the record was added). Then it must be the case that the queried range has matched the new record. The adversary can detect this, since it knows the volume of every query for the original database, as noted earlier. Since the query has matched the new record, its volume must be one more than the volume of some range in the original database. If that range is unique, once again the adversary knows this, and can deduce the queried range $[x, y]$. In that case the adversary can immediately deduce that the value of the new record must lie within $[x, y]$. A similar reasoning holds in the case that the observed range does *not* contain the new record.

As the volumes of more range queries are observed, the adversary refines its knowledge of the new record's value. We note that the previous reasoning required an assumption about a certain volume corresponding to a *unique* queried range in the original database. As both experiments and analysis will show however, this event occurs often enough that the algorithm is able to quickly home in on the value of the new record.

For space reasons, the pseudo-code of the attack (Algorithm 4) is given in Appendix B. The input of the algorithm is a set of volumes V, which should be the number of matching records (i.e. volume) of a set of observed queries; the original database counts C, where $C(k)$ is the number of records with value k; and the number of values N. The algorithm creates a table "RangeFromVol" mapping volumes to ranges for the original counts C, then proceeds to refine the set "Possible" of possible values for the new record as new volumes are observed, according to the algorithm explained in the previous paragraph. The algorithm finally outputs a guess for the value of the new record, which is simply the average of the minimum and maximum of the set of possible values obtained up to that point (this choice minimizes worst-case error). We preferred clarity in the pseudocode; many refinements are possible. Our experiments show that this simple algorithm is already quite effective.

Our analysis of the attack's query complexity appears in the full version for space reasons. Within the same analytical model as in Section 3.2, we show that assuming $R = \Omega(N^2)$, the expected number of queries for exact recovery is $O(N)$, and the expected number of queries to recover the value of the new record within ϵN is $O(1/\epsilon)$, where the constants depend on $N/\sqrt{R}$.

4.2 Experiments

We have run Algorithm 4 on the age data of patients in three hospitals of sizes within 10% of $R = 5000$, 10000 and 20000, extracted from the same HCUP data as in Section 3.3. The age data is capped at 90 in our dataset for privacy reasons, and so the number of values is $N = 91$. Thus the choices of sizes 5000, 10000 and 20000 reflect the cases where R is respectively close to $N^2/2$, N^2 and $2N^2$. Update recovery should work well in parameter regimes when the main

	Precision				
R	20%	10%	5%	2%	Exact
5000	47	79	123	229	974
10000	18	29	46	99	391
20000	11	17	27	57	191

Figure 5: Median number of queries needed to achieve the given precision in the output of Algorithm 4, for three hospitals with size within 10% of the given R.

reconstruction attack works well: around $R \geq N^2/2$. Effectiveness should degrade gracefully below that value.

If R is close to $N^2/2$ or below, it may happen in that most or all ranges that would allow to uniquely identify the value of the new record could collide with other volumes; in that case exact recovery could be very expensive or impossible. If recovery is impossible, the average number of queries required for recovery is technically infinite. Obviously, an infinite average does not reflect the fact that in practice, recovery should usually succeed with a low number of queries. For this reason we use medians instead of averages.

Results are given in Figure 5. As predicted by our model, the number of queries needed for exact recovery is of the order of magnitude of a reasonably small multiple of N, although the constant degrades when R approaches $N^2/2$. The number of queries necessary to achieve a precision ϵ does appear to behave as $O(1/\epsilon)$. Furthermore, the value of the new record can be approximated reasonably well within relatively few queries, especially for larger R: for $R = 20000$ we see that observing the volume of 27 queries suffices to recover the value of the new record within an error of 5%, i.e. within 5 years.

5 RANGE QUERY RECONSTRUCTION VIA CDF MATCHING

In Section 3, we described an attack that achieves database reconstruction from the volumes of unknown queries. Once the database has been reconstructed, the queries themselves can also be reconstructed from their volumes by matching an observed volume to the set of possible queries that have that volume. Thus, one generic approach to query reconstruction is to observe enough volumes to reconstruct the database using the previous attack, then simply match queries to volumes using the reconstructed database. In the full version of this work we confirm experimentally that this attack is very effective. A drawback of this approach is that many queries must be observed before any information is learned about any queries. If an attacker wants to learn as much as it can from a set of queries of any size, a different approach is needed.

In this section, we describe an attack that achieves query reconstruction "online", meaning the adversary can infer a set of likely underlying values for the query as soon as it observes its corresponding volume. We call this attack "CDF matching". CDF matching uses an estimate of the database distribution (below, an "auxiliary distribution") to infer the underlying (hidden) query as soon as its volume is observed. Various works [11, 20] have argued that attacks with auxiliary distributions are realistic; such distributions can come from census data [1], public employee records [11], or even copies of similar databases posted online by hackers.

First, we will describe the attack and analyze it in the setting where the adversary has full knowledge of the database distribution. Then, we will demonstrate empirically that (1) the attack reveals a substantial amount of information about queries, and (2) our analysis retains much of its predictive power even when the adversary has a poor auxiliary distribution.

Preliminaries. Before describing our attack we will state and discuss two useful technical tools. The **Kolmogorov-Smirnov (KS) distance** between CDFs F and G is $KS(F, G) = \sup_x |F(x) - G(x)|$. Let $\mathbf{I}_b$ is 1 if $b = 1$ and 0 otherwise. The **Dvoretzky-Kiefer-Wolfowitz (DKW) inequality** [6] is a Chernoff-type bound on the maximum distance between the empirical and true CDF of a distribution.

Theorem 5.1 (Dvoretzky-Kiefer-Wolfowitz). *Let π_{DB} be a distribution on $[1, N]$, and $X_1, \ldots, X_R$ be R i.i.d. samples from π_{DB}. Let $\hat{F}(i) = \frac{1}{R}\sum_{j=1}^{R} \mathbf{I}_{X_j \leq i}$ and $F(i) = \sum_{j=1}^{i} p_i$ for $i \in [1, N]$. Then* $\Pr\left[\sup_i \left|\hat{F}(i) - F(i)\right| > \epsilon\right] \leq 2e^{-2R\epsilon^2}$.

This is a useful result for several reasons. First, it implies convergence for volumes of range queries, since the volume of every query $[i, j]$ can be written $R \cdot (\hat{F}(j) - \hat{F}(i-1))$. It is also a *uniform convergence* bound, meaning its guarantees apply to all range queries simultaneously. Finally, its rate of convergence is nearly as fast as a single Chernoff bound, though it applies to many events at once.

5.1 The CDF Matching Attack

First define some notation. Let the number of records in the database be R. Let $\pi_{\text{DB}} = (p_1, \ldots, p_N)$ be a distribution over $[1, N]$, and $\mathcal{R}$ the set of all range queries on $[1, N]$. We model the database as a sequence of R i.i.d. samples $X_1, \ldots, X_R$ from π_{DB}. For any query $q = [a, b]$, define $V_{DB}(q) = \sum_{i=1}^{R} \mathbf{I}_{a \leq X_i \leq b}$ and $\Pr[q] = \sum_{\ell \in q} p_\ell$.

Let our attack be represented by the adversary $\mathcal{A}$. For any query q, $\mathcal{A}$ takes as input $\pi_{\text{DB}}, R, V_{DB}(q)$, and the confidence parameter $0 < \delta < 1$. The adversary $\mathcal{A}$ first computes a *precision* ϵ with the property that

$$\Pr\left[\sup_{q' \in \mathcal{R}} \left| \frac{V_{DB}(q')}{R} - \sum_{\ell \in q'} p_\ell \right| \leq \epsilon \right] \geq 1 - \delta$$

using the DKW inequality, as $\epsilon = \sqrt{(2 \log \frac{2}{\delta})/R}$. The adversary $\mathcal{A}$ then outputs $\hat{Q} = \left\{q' \in \mathcal{R} : \left|\Pr[q'] - \frac{V_{DB}(q)}{R}\right| \leq \epsilon\right\}$.

We will measure success on two axes: (1) *raw accuracy*, which measures whether the true query is in the candidate set $\hat{Q}$, and (2) *uncertainty reduction*, which measures the size of $\hat{Q}$ (where smaller $\hat{Q}$s reduce the attacker's uncertainty more). There are many ways to refine the CDF matching attack, such as using constraints on query volumes. One such constraint is that for two volumes $v_1 < v_2$, the query with volume v_2 cannot be contained in the query with volume v_1. Such refinements would increase reconstruction accuracy but would make the attack "offline", so we leave it to future work.

Analysis. Next we turn to analyzing the performance of the CDF matching attack. With the DKW inequality as a tool, it is straightforward to put tight analytical bounds on these two performance metrics in an ideal query reconstruction setting where the adversary has precise knowledge of the database distribution and the database is sampled i.i.d. The following theorem lower-bounds raw accuracy and can be proven via a simple application of DKW.

Theorem 5.2. *Let $\pi_{DB} = (p_1, \ldots, p_N)$ be a distribution on $[1, N]$, and let $\mathcal{R}$ be the set of all intervals of $[1, N]$. Let $DB = X_1, \ldots, X_R$ be R i.i.d. samples from π_{DB}. For a range query $q = [i, j]$ on $[1, N]$, define $\Pr[\,q\,] = \sum_{\ell=i}^{j} p_\ell$ and $V_{DB}(q) = \sum_{k=1}^{R} \mathbf{I}_{i \le X_k \le j}$. Let $\mathcal{A}$ be an adversary which on input $\pi_{DB}, R, V_{DB}(q)$ and $0 < \delta < 1$ computes $\epsilon = \sqrt{(2 \log \frac{2}{\delta})/R}$ and outputs $\hat{Q}$ as defined above. Define* Err *to be the event, over the random coins used to sample DB, $\exists q \in \mathcal{R} \mid (q \notin \mathcal{A}(\pi_{DB}, R, V_{DB}(q), \delta))$. Then $\Pr[\,\mathrm{Err}\,] \le \delta$.*

This theorem implies that an adversary that chooses its candidate set in CDF matching via the DKW inequality has perfect raw accuracy except with probability δ. Even with an inaccurate auxiliary distribution, Theorem 5.2 and other results of this form are likely to hold as long as the KS distance between the true distribution and the adversary's auxiliary distribution is low.

Next we present a theorem on the *uncertainty reduction* of the CDF matching attack. Uncertainty reduction measures the number of queries which could correspond to an observed volume. Intuitively, uncertainty about the underlying query of an observed volume is related to the number of queries whose probabilities are "close" to the real query. Since the proof is a simple application of the DKW inequality, we elide it.

Theorem 5.3. *Let $\pi_{DB} = (p_1, \ldots, p_N)$ be a distribution on $[1, N]$, and let $\mathcal{R}$ be the set of all intervals of $[1, N]$. Let $DB = X_1, \ldots, X_R$ be R i.i.d. samples from π_{DB}. For a range query $q = [i, j]$ on $[1, N]$, define $\Pr[\,q\,] = \sum_{\ell=i}^{j} p_\ell$ and $V_{DB}(q) = \sum_{k=1}^{R} \mathbf{I}_{i \le X_k \le j}$. For any query q, $0 < \delta < 1$, and $\epsilon = \sqrt{(2 \log \frac{2}{\delta})/R}$ define $C_q = \{q' \in \mathcal{R} \mid |\Pr[\,q\,] - \Pr[\,q'\,]| \le 2\epsilon\}$ and $CS_q = \{q' \in \mathcal{R} \mid |\Pr[\,q'\,] - (V_{DB}(q)/R)| \le \epsilon\}$ (Note that the C_q is fixed by π_{DB}, δ, R while CS_q is a random variable.) Define* CS *to be the event, over the random coins used to sample DB, $\bigcup_{q \in \mathcal{R}} |CS_q| > |C_q|$. Then $\Pr[\,\mathrm{CS}\,] \le \delta$.*

This theorem is useful because it relates the size of the candidate set for a query to the number of other queries whose expected volumes are close in probability. Further, we can quantify the rate at which the candidate set gets smaller as the number of records increases.

5.2 Experimental Results

We performed two types of experiments to assess the risk of query reconstruction. The first type assumes the exact count of each element in the database is known, either because the database was stolen or the attacker observed enough queries to run the clique-finding algorithm described above. For space reasons, a description of the exact procedure and the results appear in the full version of this paper.

The second type of experiment evaluates the CDF matching algorithm using the attributes described in Figure 7 of Appendix C. In each experiment, we took the individual hospital records for that attribute for a particular year's HCUP data to be the targets of the attack. We used the aggregate counts of a different year of HCUP data as the auxiliary data. We ran experiments with different combinations of auxiliary data and target hospitals for the HCUP years 2004, 2008, 2009 and 2013. Surprisingly, both the performance of the experiments and the median KS distance between the auxiliary distribution and the target hospitals varied only a small amount between different experiments for an attribute, so for simplicity of exposition we will only present one experiment for each.

An individual experiment performs the following steps: first, we compute the number of records R in the target hospital. Then we compute the epsilon given by the DKW inequality for R and $\delta = 0.05$. Then for each query in the target hospital, we compute the set of candidate queries $\hat{Q}$ as described above. Figure 6 shows the median raw recovery rate (i.e. the median fraction of times the correct query is in $\hat{Q}$) broken down by the sizes of the sets $\hat{Q}$. The set sizes are relative to the total number of queries for each attribute (given in Figure 7); to save space we omit converting the percentages to absolute sizes. Roughly, the total height of each bar is the median fraction of correct predictions (of any size) and the different patterns on each bar report how much reduction in uncertainty each correct prediction gives the adversary (where a *smaller* number means the size of $\hat{Q}$ is smaller, and the adversary's uncertainty is reduced more). With precise knowledge of the database distribution and i.i.d. samples, the median raw recovery rate would be 100% except with probability 0.05. With no knowledge at all, the "baseline guessing" attack would simply set $\hat{Q}$ to be all possible queries. On this graph, this would be a bar with the 100% pattern going from 0.0 to 1.0.

Discussion. The raw recovery rate varied widely between different attributes. For the two largest attributes (AGEDAY and LOS) almost every set $\hat{Q}$ contained the correct query. However, both attributes have an extremely skewed distribution, so for almost all queries $\hat{Q}$ contained almost every possible query. Thus, the "reconstruction" achieved for most queries is not better than baseline guessing. The attack performed well on AMONTH, and there the sets $\hat{Q}$ were much smaller—over 30% of the recovered queries had $|\hat{Q}| \le 8$. The auxiliary data was quite good for AMONTH: the median KS distance was only 0.02.

The results for NDX, NPR, and NCHRONIC are more surprising. All three had relatively large median KS distances, but a substantial fraction (around 15%) of all queries were correctly recovered *and* had small $|\hat{Q}|$. For NDX and NPR, around 15% of queries were recovered and had $|\hat{Q}| \le 14$, and for NCHRONIC around 10% of queries had $|\hat{Q}| \le 35$. The overall recovery rate was high as well. AGE also had many correctly-recovered queries with small $\hat{Q}$, despite a high KS distance. In fact, in more than 80% of hospitals there were correctly-recovered queries with $|\hat{Q}| \le 15$, which corresponds to only 0.4% of possible range queries! Despite having a poor auxiliary distribution, for all these attributes the attack was able to recover fine-grained information about many queries. Further, the analysis (in particular Theorem 5.2), which formally only holds when the auxiliary distribution is nearly perfect, is still partially predictive for accuracy in a noisy setting.

The conclusions we draw from these experiments are twofold: (1) simple query reconstruction attacks can reveal fine-grained

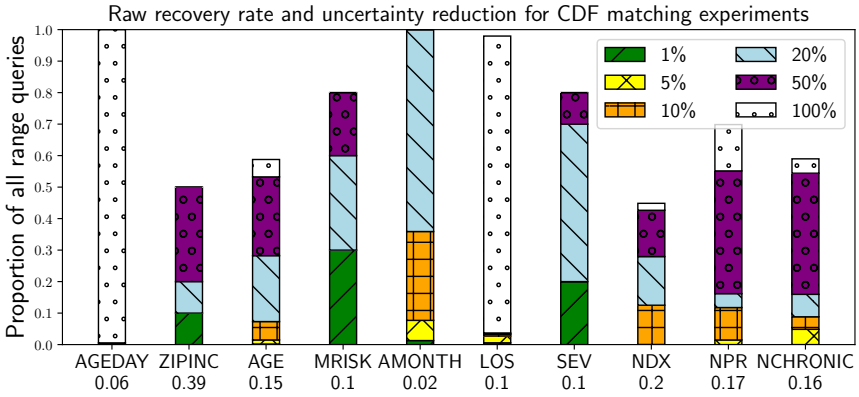


Figure 6: Results of CDF matching experiments on HCUP attributes. The number appearing below each attribute is the median across all hospitals of the KS distance between an individual hospital and the auxiliary distribution. The percentage corresponding to each bar pattern is the proportion of queries in the set $\hat{Q}$ for the correct predictions; thus, smaller numbers are better.

information about queries and damage privacy even in practical settings and with poor auxiliary data, and (2) idealized models of these attacks proven under seemingly strong assumptions (such as perfect auxiliary data or i.i.d. samples) maintain much of their predictive power when these assumptions are violated.

Database reconstruction. If enough queries are reconstructed with high accuracy, it is possible to reconstruct the database as well. If we write each query $q = [a, b]$ as a 0-1 row vector where $q_i = 1$ if $a \leq i \leq b$ and zero otherwise, the database DB (a vector with N components whose sum is R) is the solution to the system of linear equations $\boldsymbol{Q} \cdot DB = \vec{v}$ where $\boldsymbol{Q}$ is a matrix of row vectors for each query, and $\vec{v}_i$ is the volume of the ith query. Of course, one can use the attack from Section 3 to reconstruct the database; the advantage of this approach is that it requires far fewer queries. We leave a more detailed treatment to future work.

6 COUNTERMEASURES

In this section we briefly discuss some possible countermeasures to our attacks. There are two basic kinds of countermeasures: client processing and adding noise.

Volume information can be hidden if the client does some additional processing of queries and results. If instead of issuing a single query, the client batches several queries together, the volume of any individual query will not be revealed. If queries are infrequent, this could incur a high latency penalty. Another approach is putting a lower limit on the width of a range query. If the client wants to query a small range, it queries a larger range and filters the unneeded results locally. This incurs bandwidth overhead, but may be feasible in some settings. The database could be bucketed, meaning records which are close in value are treated as one logical "value" for the purposes of retrieval [13]. Bucketing would not generally prevent reconstruction, but would ensure the exact counts of individual elements are not revealed.

Adding noise to the volumes can be done by adding dummy records to the database, incurring server storage overhead. It seems inherent that the security benefit is directly related to the storage overhead: a small number of dummy records (yielding low storage overhead) will give little or no security benefit. One principled way of adding dummy records is using differential privacy (DP), as suggested by KKNO in a follow-up work [15]. Rather than querying the database directly, they query the output of a DP mechanism for range queries. Intuitively, the DP mechanism prevents reconstruction of the exact count of every element in the database. Crucially, their guarantees do not extend to query reconstruction: while query reconstruction *should* be less accurate, no formal guarantee precludes accurate query reconstruction. Since a thorough examination of DP countermeasures would be quite involved, we leave it to future work.

7 RELATED WORK

Aside from KKNO, there are two recent works on reconstruction attacks which are related to ours. The first is by Cash et al. [3], who present an attack for revealing keyword search queries on natural-language documents based on the number of results returned. Their sole attack in the volume-only setting requires perfect knowledge of the documents in the database and simply matches an observed volume with the query having that count. Our query reconstruction attack with exact counts (discussed in Section 5) can be seen as a version of their count attack.

The other recent paper related to our attacks is by Lacharité *et al.* [16]. Their auxiliary data attack is similar in some ways to the CDF matching attack in Section 5. They target full reconstruction, assuming both access pattern and rank leakage, but do not provide a formal analysis. In contrast, our CDF matching attack targets query reconstruction with fewer assumptions. Moreover, it is accompanied by an analysis which gives tight theoretical guarantees and maintains its predictive ability even if the auxiliary data is inaccurate.

In the security community, communication volume and other traffic features like packet timings have long been used to perform traffic analysis and website fingerprinting attacks. For example, Wright et al. [24] recovered spoken phrases from encrypted VoIP traffic by training a model on packet sizes and timings. Both the settings and goals of these works are distinct from ours; in particular they rely on information that is not available in our setting.

Some countermeasures for communication volume leakage exist. For example, IPSec has an optional "Traffic Flow Confidentiality" mode that adds padding. TLS 1.3 and SSH also allow packets to be padded. These countermeasures are not widely used in practice, both because they are usually too expensive to deploy in large systems and because prior work [7] has shown the overall usefulness of these countermeasures is quite low.

In a recent follow-up [15] to their reconstruction attacks, KKNO combine ORAM and differential privacy with the goal of preventing database reconstruction attacks based on either access pattern or communication volume. We discuss this work in Section 6.

8 CONCLUSIONS AND FUTURE DIRECTIONS

In this work we demonstrate practical reconstruction attacks which use only volume leakage. In the context of encrypted databases, it is worth noting that while ORAM protects against attacks that require access pattern leakage, it remains vulnerable to volume attacks. Given the rich volume information that can be recovered from a database snapshot [10], one unavoidable and surprising conclusion of this work is that ORAM by itself is insufficient to argue security even against reconstruction attacks carried out by a snapshot attacker. An empirical study of whether our attacks can be carried out using only a database snapshot is an interesting direction for future work.

ACKNOWLEDGMENTS

The first author was supported by NSF Graduate Research Fellowship DGE-1650441. Portions of this work were written while the first author was visiting Royal Holloway, University of London. The second author was supported by the European Union's Horizon 2020 research and innovation programme under grant agreement H2020-MSCA-ITN-2014-643161 ECRYPT-NET. The third and fourth authors were supported by EPSRC Grant EP/M013472/1.

REFERENCES

[1] Vincent Bindschaedler, Paul Grubbs, David Cash, Thomas Ristenpart, and Vitaly Shmatikov. 2018. The Tao of Inference in Privacy-Protected Databases. In *PVLDB*.
[2] Coen Bron and Joep Kerbosch. 1973. Algorithm 457: Finding All Cliques of an Undirected Graph. *Commun. ACM* (1973).
[3] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. 2015. Leakage-Abuse Attacks Against Searchable Encryption. In *ACM CCS 15*, Indrajit Ray, Ninghui Li, and Christopher Kruegel: (Eds.). ACM Press, 668–679.
[4] David Cash, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. 2013. Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries. In *CRYPTO 2013, Part I (LNCS)*, Ran Canetti and Juan A. Garay (Eds.), Vol. 8042. Springer, Heidelberg, 353–373. https://doi.org/10.1007/978-3-642-40041-4_20
[5] Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. 2006. Searchable symmetric encryption: improved definitions and efficient constructions. In *ACM CCS 06*, Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati (Eds.). ACM Press, 79–88.
[6] Aryeh Dvoretzky, Jack Kiefer, and Jacob Wolfowitz. 1956. Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. *The Annals of Mathematical Statistics* (1956).
[7] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. 2012. Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail. In *2012 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 332–346.
[8] Philippe Flajolet, Daniele Gardy, and Loÿs Thimonier. 1992. Birthday paradox, coupon collectors, caching algorithms and self-organizing search. *Discrete Applied Mathematics* (1992).
[9] Oded Goldreich. 1987. Towards a Theory of Software Protection and Simulation by Oblivious RAMs. In *19th ACM STOC*, Alfred Aho (Ed.). ACM Press, 182–194.
[10] Paul Grubbs, Thomas Ristenpart, and Vitaly Shmatikov. 2017. Why your encrypted database is not secure. In *HotOS*.
[11] Paul Grubbs, Kevin Sekniqi, Vincent Bindschaedler, Muhammad Naveed, and Thomas Ristenpart. 2017. Leakage-Abuse Attacks against Order-Revealing Encryption. In *2017 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 655–672.
[12] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*.
[13] Bijit Hore, Sharad Mehrotra, and Gene Tsudik. 2004. A privacy-preserving index for range queries. In *VLDB*.
[14] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. 2016. Generic Attacks on Secure Outsourced Databases. In *ACM CCS 16*, Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi (Eds.). ACM Press, 1329–1340.
[15] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. 2017. Accessing data while preserving privacy. *arXiv 1706.01552* (2017).
[16] Marie-Sarah Lacharité, Brice Minaud, and Kenneth G. Paterson. 2018. Improved Reconstruction Attacks on Encrypted Data Using Range Query Leakage. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society Press, 1–18. doi.ieeecomputersociety.org/10.1109/SP.2018.00002
[17] Michael Luby. 1985. A Simple Parallel Algorithm for the Maximal Independent Set Problem. In *17th ACM STOC*. ACM Press, 1–10.
[18] Michael Mitzenmacher and Eli Upfal. 2005. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge university press.
[19] J. W. Moon and L. Moser. 1965. On cliques in graphs. *Israel Journal of Mathematics* (1965).
[20] Muhammad Naveed, Seny Kamara, and Charles V. Wright. 2015. Inference Attacks on Property-Preserving Encrypted Databases. In *ACM CCS 15*, Indrajit Ray, Ninghui Li, and Christopher Kruegel: (Eds.). ACM Press, 644–655.
[21] Tiago P. Peixoto. 2014. The graph-tool python library. (2014). https://doi.org/10.6084/m9.figshare.1164194
[22] Roei Schuster, Vitaly Shmatikov, and Eran Tromer. 2017. Beauty and the Burst: Remote Identification of Encrypted Video Streams. In *USENIX Security*.
[23] Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher W. Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. 2013. Path ORAM: an extremely simple oblivious RAM protocol. In *ACM CCS 13*, Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung (Eds.). ACM Press, 299–310.
[24] Charles V Wright, Lucas Ballard, Scott E Coull, Fabian Monrose, and Gerald M Masson. 2008. Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversations. In *IEEE S&P*.
[25] George Kingsley Zipf. 1935. The psychobiology of language. (1935).

A DETAILED DESCRIPTION OF KKNO RECONSTRUCTION ATTACKS

In this appendix we reproduce, for completeness, the reconstruction attacks presented in [14]. We also give some evidence that it is difficult (or perhaps even impossible) to adapt to non-uniform query distributions, and conclude by noting some surprising limitations of KKNO's attack.

A.1 KKNO's Factorization Attack

Suppose the database contains R records with values $val_1 \leq val_2 \leq \ldots \leq val_R$ in $\{1, \ldots, N\}$. For ease of notation, define $val_0 := 0$ and $val_{R+1} := N + 1$. Define the *distance* d_k between the kth and $(k+1)$st records as $d_k := val_{k+1} - val_k$. Let u_k be the number of distinct queries matching k records, for any $0 \leq k \leq R$. (Note that since there are only $N(N+1)/2$ distinct queries, at most this many u_k's can be non-zero.) The u_k's then satisfy the following equations:

$$\begin{aligned} u_R &= d_0 \cdot d_R \\ u_{R-1} &= d_0 \cdot d_{R-1} + d_1 \cdot d_R \\ &\vdots \\ u_{R-m} &= \sum_{k=0}^{m} d_k \cdot d_{R-(m-k)} \end{aligned}$$

$$\vdots$$
$$u_1 = d_0 \cdot d_1 + d_1 \cdot d_2 + \ldots + d_{R-1} \cdot d_R$$
$$u_0 = 1/2 \left(\sum_{k=0}^{R} {d_k}^2 - (N+1) \right)$$

The key observation from KKNO's work is that if queries are uniformly distributed, then it is possible to determine the u_k's by observing enough queries, after which it is possible to construct a polynomial that can be factored into two polynomials whose coefficients are the d_k's. That polynomial is

$$F(x) = u_R + \ldots + u_1 \cdot x^{R-1} + \hat{u}_0 \cdot x^R + u_1 \cdot x^{R+1} + \ldots + u_R \cdot x^{2R},$$

where the coefficient of x^R is not u_0, but $\hat{u}_0 \coloneqq 2 \cdot u_0 + N + 1$. The polynomial can then be factored into two degree-R polynomials, specifically, $F(x) = d(x) \cdot d^r(x)$, where

$$d(x) = d_0 + d_1 \cdot x + \ldots + d_R \cdot x^R, \text{ and}$$
$$d^r(x) = d_R + d_{R-1} \cdot x + \ldots + d_0 \cdot x^R.$$

Polynomial F may not have a unique factorisation into factors of this form, in which case KKNO's algorithm picks one of the possibilities arbitrarily.

The first step of the attack is to observe enough queries to determine the u_k's. Let c_k be the number of queries (out of the observed Q queries) that have volume k, for any k between 0 and R. Then as the number of queries Q grows, the quantity c_k/Q gets arbitrarily close to $u_k \cdot \frac{1}{N(N+1)/2}$, and therefore it is possible to solve for the u_k's and proceed with constructing and factorizing $F(x)$. KKNO show that the approach we just sketched correctly recovers all u_k's after $\Omega(N^4 \log N)$ queries, except with inverse polynomial (in N) probability. Note that this attack can also cope with some d_i values being zero.

A.2 KKNO for Non-Uniform Query Distributions

The assumption that the query distribution is uniform is inherent to KKNO's algorithm; without such an assumption, it is not clear how to determine the u_k's from the c_k's. In fact, with non-uniform query distributions, it is sometimes not possible to uniquely determine the u_k's. Consider the following example with $N = 3$, $R = 3$, and query distribution as follows:

$$Pr([1,1]) = 1/4 \qquad Pr([1,2]) = 1/6 \qquad Pr([1,3]) = 1/6$$
$$Pr([2,2]) = 1/8 \qquad Pr([2,3]) = 1/8$$
$$Pr([3,3]) = 1/6.$$

Suppose that after sufficiently many queries Q have been observed, the counts are as follows:

$$c_0/Q = 1/6 \qquad c_1/Q = 1/4 \qquad c_2/Q = 1/4 \qquad c_3/Q = 1/3.$$

In this case, it is impossible to distinguish whether the element counts are $\{1, 2, 0\}$ (or the reflection $\{0, 2, 1\}$) or $\{2, 1, 0\}$ (or the reflection $\{0, 1, 2\}$). Note that with a uniform query distribution, KKNO's algorithm would have succeeded in both cases since the corresponding polynomials factor into unique pairs of degree-3 polynomials:

$$F_1(x) = 2 + 2x + x^2 + 6x^3 + x^4 + 2x^5 + 2x^6$$
$$= (x^3 + x^2 + 2)(2x^3 + x + 1), \text{ and}$$
$$F_2(x) = 2 + x + 2x^2 + 6x^3 + 2x^4 + x^5 + 2x^6$$
$$= (x^3 + x + 2)(2x^3 + x^2 + 1).$$

Impossibility of perfectly correct reconstruction. Interestingly, even under KKNO's precise assumptions about the query distribution, there are databases that cannot be reconstructed (even up to reflection). This is because the function that takes a database and outputs its u_i values *is not injective*—there are databases on which this function collides! This implies, surprisingly, that the most natural notion of correctness for reconstruction attacks (that for a fixed database, as the number of observed queries goes to infinity, the attack should be able to reconstruct up to reflection with probability 1) is impossible to achieve when the query distribution is uniform, because there are some distinct databases that generate the same set of volumes. We implemented a simple brute-force experiment that computes the u_i values and runs KKNO's attack on every possible database where $N = 17$ and with $2 \leq R \leq 15$. With $R = 15$ there are just over 300 million possible databases, and KKNO failed due to u_i collision on 49,000 of them. Here are two databases which have the same u_i values:

$$\{0, 0, 0, 3, 1, 0, 0, 1, 0, 1, 0, 6, 1, 0, 1, 0, 1\}$$
$$\{1, 1, 2, 1, 1, 0, 2, 2, 2, 2, 0, 0, 0, 1, 0, 0, 0\}$$

We found u_i collisions on dense databases as well. Understanding this phenomenon seems to be a difficult but interesting combinatorics question which we leave to future work. The theory of Golomb rulers may give some insight here—observe that two databases whose elementary volumes are Golomb rulers (i.e. sets of numbers that generate every possible value below some threshold via pairwise subtraction) can never be distinguished from volume alone.

B DESCRIPTIONS OF ALGORITHMS AND CORRECTNESS PROOFS

This section contains pseudocode for several algorithms used in the main body, as well as proofs of their correctness.

In Algorithm 2, the Find_Maximal_Cliques subroutine called on line 14, which returns all maximal cliques in the subgraph induced by CAND_{nn}, is not specified: it can be implemented using existing clique-finding algorithms.

In Algorithm 3, we provide more practical variants of some subroutines from Algorithm 2. We also modify the algorithm to return all solutions (when possible), not only minimal ones. Specifically, we replaced Min_Subcliques with All_Subcliques_P as defined starting on line 15 in Algorithm 3. If it may be impractical to enumerate the subcliques whose sizes are in the right range, we do not return any subcliques – the final solutions will be incomplete.

Lemma B.1 (Correctness of Alg. 1). *Let DB be a database with at most N different values, let V be the set of all possible range query volumes, and let V_{elem} be the set of elementary range volumes that contains the minimum complemented volume. Then, after running*

Algorithm 1 on (N, V) to obtain the sets V_{nec} and V_{cand} of necessary and candidate nodes, we have

$$V_{\text{nec}} \subseteq V_{\text{elem}} \subseteq V_{\text{cand}}.$$

Proof. We show that $V_{\text{nec}} \subseteq V_{\text{elem}} \subseteq V_{\text{cand}}$ holds throughout Alg. 1. After line 11, we have $V_{\text{nec}} \subseteq V_{\text{elem}}$ since $R = \text{vol}([1, N])$ is in V_{elem}, and $v_{\min}$ is in V_{elem} by design. After line 12, we have $V_{\text{elem}} \subseteq V_{\text{cand}}$ since all elementary volumes are complemented. To complete the proof, we show that if $V_{\text{nec}} \subseteq V_{\text{elem}} \subseteq V_{\text{cand}}$, then (i) Augment_NEC $(V_{\text{cand}}, V_{\text{nec}}) \subseteq V_{\text{elem}}$, and (ii) $V_{\text{elem}} \subseteq$ Reduce_CAND$(V_{\text{cand}}, V_{\text{nec}})$.

First, consider the three ways in which Augment_NEC can add elements to V_{nec}.

- (line 23) If $|V_{\text{cand}}| = N_{\min}$ and $V_{\text{elem}} \subseteq V_{\text{cand}}$, then clearly $V_{\text{elem}} = V_{\text{cand}}$ since $|V_{\text{elem}}| \geq N_{\min}$.
- (line 26) Let e be a non-complemented volume. Since $V_{\text{elem}} \subseteq V_{\text{cand}}$, we know that every volume, including e, arises as a node or an edge (or both) in the graph induced by V_{cand}. The volume e has no complement, so it must arise as an edge, i.e., as the absolute difference of two volumes in V_{cand}. If all such edges are incident to one node V_{cand}, then it must be in V_{elem}.
- (line 30) Let v be a non-necessary complemented volume in V_{cand}. Every volume, including v, arises as a node or an edge (or both) in the graph induced by V_{cand}. If the volume v arises only as itself and maybe edges incident to itself, then it must be in V_{elem}.

Next, consider Reduce_CAND. Let v be a non-necessary complemented volume in V_{cand}. Since $V_{\text{nec}} \subseteq V_{\text{elem}}$, and the volumes in V_{elem} are all adjacent to each other, any node that is not adjacent to a subset of volumes in V_{elem} cannot be in V_{elem}. □

The main Get_Elem_Volumes procedure uses a few subroutines. Gen_All_Volumes checks whether a subset of nodes generates all volumes in a given set (and perhaps other volumes). Gen_Exact_Volumes additionally checks that only the volumes in the given set are generated. Min_Subcliques takes a clique V_k that generates all volumes of a given set V_{all} and finds the minimal subclique(s) of V_k that generate exactly the volumes in V_{all}. Here, "minimal" refers to no strict subset of them generating all volumes in V_{all}.

Lemma B.2 (Correctness of Alg. 2). *Let DB be a database of elements with N possible different values, let V be the set of all range query volumes, and let V_{elem} be the set of elementary range volumes that contains the minimum complemented volume. Suppose we are given two sets V_{nec} and V_{cand} such that $V_{\text{nec}} \subseteq V_{\text{elem}} \subseteq V_{\text{cand}}$. Then, after running Algorithm 2 on $(N, V_{\text{cand}}, V_{\text{nec}}, V)$ to obtain the set solutions, we have*

- *$V_{\text{elem}} \in$ solutions if $0 \notin V$ (the data is dense), or*
- *$V_{\text{elem}} \supseteq s$ for at least one $s \in$ solutions (if the data is sparse).*

Proof. First, if the data is dense ($0 \notin V$), then clearly the number of elementary volumes is $N_{\min} = N_{\max} = N$. If the data is sparse, there must be at least $N_{\min} \stackrel{\text{def}}{=} \left\lceil -0.5 + 0.5 \cdot \sqrt{1 + 8 \cdot |V|} \right\rceil$ elementary volumes, otherwise there would be strictly fewer than $|V|$ range query volumes. There can be at most $N - 1$ because the occurrence of the volume 0 means at least one of the N values did not appear in *DB*, and at most $|V_{\text{cand}}|$ because it is known that $V_{\text{elem}} \subseteq V_{\text{cand}}$.

Now consider the graph $G = (V_{\text{cand}}, E)$ with edge set

$$E \stackrel{\text{def}}{=} \{(v_1, v_2) \in V_{\text{cand}} \times V_{\text{cand}} : |v_2 - v_1| \in V \setminus \{0\}\}$$

and the induced subgraph $G_{\text{nn}} \stackrel{\text{def}}{=} G(V_{\text{cand}} \setminus V_{\text{nec}})$. Since the subgraph induced by V_{elem} is a clique in G, the subgraph induced by $V_{\text{elem}} \setminus V_{\text{nec}}$ will also be a clique in G_{nn}. Therefore, at least one of the maximal cliques in G_{nn} output by Find_Maximal_Cliques (line 14), say V_k^*, will have $V_{\text{elem}} \setminus V_{\text{nec}}$ as a subclique. The size of V_k^* must be at least $N_{\min} - |V_{\text{nec}}|$, so the algorithm will proceed to line 19 in this iteration. Since $V_{\text{elem}} \subseteq \{V_{\text{nec}} \cup V_k^*\}$ generates all volumes in V (and maybe others), *solutions* will be updated to include the output of Min_Subcliques (line 20).

Since $V_{\text{elem}} \setminus V_{\text{nec}}$ is a subset of V_k^*, it will arise as a subclique on line 39 of Min_Subcliques. V_{elem} generates all volumes in V and no others, so the algorithm will proceed to line 41. If the data is dense, then *subcliques* may contain only sets of size $N = N_{\min} = N_{\max}$, so V_{elem} cannot be a superset of any other element in *subcliques*, so it will be added to this set. If the data is not dense, however, then either (i) there is already a strict subset of V_{elem} in *subcliques* that generates exactly the volumes in V, or (ii) there is no such set,

Algorithm 1 Graph pre-processing: finding a smaller subgraph.

```
1:  procedure Graph_Preprocessing(N, V)
2:      R ← max{V}
3:      if 0 ∈ V then
4:          V ← V \ {0}
5:          N_min ← ⌈−0.5 + 0.5 · √(1 + 8 · |V|)⌉
6:      else
7:          N_min ← N
8:      V_comp ← {v ∈ V : R − v ∈ V} ∪ {R}
9:      V̄_comp ← V \ V_comp
10:     v_min ← min{V_comp}                          ▷ vol([1, 1]) or vol([N, N])
11:     V_nec ← {v_min, R}
12:     V_cand ← V_comp
13:     all_processed ← FALSE
14:     while not all_processed do
15:         V*_nec ← Augment_NEC(V_cand, V_nec, V̄_comp, N_min)
16:         V*_cand ← Reduce_CAND(V_cand, V*_nec, V)
17:         if V*_cand = V_cand and V*_nec = V_nec then
18:             all_processed ← TRUE
19:         V_nec ← V*_nec
20:         V_cand ← V*_cand
21:     return V_cand, V_nec

22: procedure Augment_NEC(V_cand, V_nec, V̄_comp, N_min)
23:     if |V_cand| = N_min then
24:         V_nec ← V_cand
25:         return V_nec
26:     for each e ∈ V̄_comp :
27:         for each v ∈ V_cand \ V_nec :
28:             if ∄(w, w′) ∈ (V_cand \ {v})² : |w − w′| = e then
29:                 V_nec ← V_nec ∪ {v}
30:     for each v ∈ V_cand \ V_nec :
31:         if ∄(w, w′) ∈ (V_cand \ {v})² : |w − w′| = v then
32:             V_nec ← V_nec ∪ {v}
33:     return V_nec

34: procedure Reduce_CAND(V_cand, V_nec, V)
35:     for each v ∈ V_cand \ V_nec :
36:         for each v_nec ∈ V_nec :
37:             if |v − v_nec| ∉ V then
38:                 V_cand ← V_cand \ {v}
39:     return V_cand
```

Algorithm 2 Recovering elementary volumes via clique-finding.

procedure GET_ELEM_VOLUMES(N, V_{cand}, V_{nec}, V)
  if $|V_{\text{cand}}| = |V_{\text{nec}}|$ **then** ▷ Pre-processing success
    return V_{nec}
  if $0 \in V$ **then**
    $V \leftarrow V \setminus \{0\}$
    $N_{\min} \leftarrow \left\lceil -0.5 + 0.5 \cdot \sqrt{1 + 8 \cdot |V|} \right\rceil$
    $N_{\max} \leftarrow \min\{N - 1, |V_{\text{cand}}|\}$
  else
    $N_{\min} \leftarrow N$
    $N_{\max} \leftarrow N$
  $M_{\min} \leftarrow \max\{0, N_{\min} - |V_{\text{nec}}|\}$
  $M_{\max} \leftarrow N_{\max} - V_{\text{nec}}$
  $\text{CAND}_{\text{nn}} \leftarrow V_{\text{cand}} \setminus V_{\text{nec}}$ ▷ Non-necessary candidate elem. volumes
  cliques $\leftarrow$ FIND_MAXIMAL_CLIQUES(CAND_{nn})
  solutions $\leftarrow \{\}$
  for all $V_k \in$ *cliques* **do**
    if $|V_k| < M_{\min}$ **then**
      continue
    if GEN_ALL_VOLUMES($V_{\text{nec}} \cup V_k$, V) **then**
      solutions $\leftarrow$ *solutions* $\cup$ MIN_SUBCLIQUES(V_k, V, $M_{\min}$, $M_{\max}$, V_{nec})
  return *solutions*

procedure GEN_ALL_VOLUMES(V_{nodes}, V_{all})
  for all $v \in V_{\text{all}}$ **do**
    if $\nexists\, (v_1, v_2) \in V_{\text{nodes}} \times V_{\text{nodes}} : |v_2 - v_1| = v$ **then**
      if $v \notin V_{\text{all}}$ **then**
        return FALSE
  return TRUE

procedure GEN_EXACT_VOLUMES(V_{nodes}, V_{all})
  if $V_{\text{nodes}} \subseteq V_{\text{all}}$ **and** GEN_ALL_VOLUMES(V_{nodes}, V_{all}) **then**
    for all $(v_1, v_2) \in V_{\text{nodes}} \times V_{\text{nodes}}$ **do**
      if $|v_2 - v_1| \notin V_{\text{all}}$ **then**
        return FALSE
    return TRUE
  else
    return FALSE

procedure MIN_SUBCLIQUES(V_k, V_{all}, $m_{\min}$, $m_{\max}$, V_{nec})
  subcliques $\leftarrow \{\}$
  for all $m \in \{m_{\min}, \ldots, \min\{m_{\max}, |V_k|\}\}$ **do**
    for all $V_{sk} \in m$-subsets(V_k) **do**
      if GEN_EXACT_VOLUMES($V_{\text{nec}} \cup V_{sk}$, V_{all}) **then**
        if $\nexists\, V'_k \in$ *subcliques* $: V'_k \subset \{V_{\text{nec}} \cup V_{sk}\}$ **then**
          subcliques $\leftarrow$ *subcliques* $\cup \{V_{\text{nec}} \cup V_{sk}\}$
  return *subcliques*

Algorithm 3 Practical, probabilistic subroutines for Alg. 2.

procedure FIND_MAXIMAL_CLIQUES_P(CAND_{nn}, $M_{\min}$, V_{all})
  cliques $\leftarrow \{\}$
  if $|\text{CAND}_{\text{nn}}| \leq 20$ **then**
    cliques $\leftarrow$ FIND_MAXIMAL_CLIQUES(CAND_{nn}) ▷ NetworkX
  else
    for $i \in \{1, \ldots, 1000\}$ **do**
      a_clique $\leftarrow$ FIND_A_MAXIMAL_CLIQUE(CAND_{nn}) ▷ graph-tool
      if $|a\_clique| \geq M_{\min}$ **then**
        cliques $\leftarrow$ *cliques* $\cup \{a\_clique\}$
    if *cliques* $= \{\}$ **then**
      return FAILURE ▷ All sampled cliques too small
    if $\nexists\, V_k \in$ *cliques* : GEN_ALL_VOLUMES($V_{\text{nec}} \cup V_k$, V_{all}) **then**
      return FAILURE ▷ No sampled clique gen. all volumes
  return *cliques*

procedure ALL_SUBCLIQUES_P(V_k, V_{all}, $m_{\min}$, $m_{\max}$, V_{nec})
  subcliques $\leftarrow \{\}$
  if $\sum_{m=m_{\min}}^{m_{\max}} \binom{|V_k|}{m} \leq 2000$ **then**
    for all $m \in \{m_{\min}, \ldots, \min\{m_{\max}, |V_k|\}\}$ **do**
      for all $V_{sk} \in m$-subsets(V_k) **do**
        if GEN_EXACT_VOLUMES($V_{\text{nec}} \cup V_{sk}$, V_{all}) **then**
          subcliques $\leftarrow$ *subcliques* $\cup \{V_{\text{nec}} \cup V_{sk}\}$
  return *subcliques*

and V_{elem} is added to *subcliques*. In all cases, any element added to *subcliques* in MIN_SUBCLIQUES will form part of the *solutions* output by GET_ELEM_VOLUMES, completing the proof. □

Algorithm 4 Update recovery attack.

procedure UPDATE_RECOVERY(V,C,N)
  RangeFromVol $\leftarrow$ empty map
  for $x \in [1, N]$ **do**
    for $y \in [x, N]$ **do**
      $v \leftarrow \sum_{x \leq k \leq y} C(k)$
      if RangeFromVol(v) is undefined **then**
        RangeFromVol(v) $\leftarrow [x, y]$
      else
        RangeFromVol(v) $\leftarrow \perp$
  Possible $\leftarrow$ [1, N] ▷ Set of possible values
  for $v \in V$ **do** ▷ Iterate through observed volumes
    if RangeFromVol($v - 1$) $= [x, y]$ and RangeFromVol(v) is undefined **then**
      Possible = Possible $\cap$ [x, y]
    if RangeFromVol(v) $= [x, y]$ and RangeFromVol($v - 1$) is undefined **then**
      Possible = Possible $\setminus$ [x, y]
  return (min(Possible) + max(Possible))/2

C DESCRIPTION OF EXPERIMENTAL DATA

In this appendix we will describe the HCUP datasets used in our experiments, as well as the steps we took to extract and process the data. First, we will provide some background on the data. The Agency for Healthcare Research and Quality (AHRQ) is a US government agency which collects a vast amount of data on the American healthcare industry. One of their core projects is the Healthcare Cost and Utilization Project (HCUP), which tracks how healthcare is used and paid for by different demographic groups. Within HCUP there are samples of different types taken every year and made available to researchers. We use the National Inpatient Sample (NIS) as our source of experimental data in this paper. Below and in the main body, when we refer to "HCUP data" we mean the NIS. The NIS is processed in a de-identifying way that protects patient privacy. **We did not attempt to deanonymize any of the data, nor are our attacks designed to deanonymize medical data.** All authors underwent the HCUP Data Use Agreement training and submitted signed Data Use Agreements to the HCUP Central Distributor.

Figure 7 contains information about the number of hospitals contained in each year's HCUP release and the minimum, maximum, and quartiles for the number of records per hospital. There is not too much year-to-year variation in the number of records per hospital for these years, which makes sense considering that until 2012 the HCUP data was collected as a random sample from all hospitals in the USA. This provides evidence that our experiments would be predictive of our attacks' performance (were they carried out on a real hospital database). In 2012, the sampling methodology for HCUP changed—more recent HCUP data is collected using a random sample of *patients* instead of hospitals. We used the 2013 HCUP data (which contains about seven million patient records) in our query reconstruction experiment in Section 5 as a source of auxiliary data. Despite the 2013 auxiliary data being a somewhat poorer estimate of per-hospital distributions for earlier years, our query reconstruction attack still performed well with the 2013 auxiliary data (even when attacking 2004 hospitals!).

Attribute-specific processing. Information about the different types of attributes is provided in Figure 7. Every year the AHRQ prescribes a format and size for each attribute collected in the various samples. In extracting per-attribute experimental data from HCUP we faced three main complications: (1) hospitals do not generally abide by these prescriptions, (2) the prescribed formats

change from year to year, and (3) not all attributes exist in all years of HCUP data. We will describe how we address each of these complications in turn.

Hospitals are strongly encouraged (but not required) to report data in the format dictated by the AHRQ, and some hospitals choose to report their data in incorrect or outdated formats. The AHRQ corrects some of these mistakes before making samples available publicly, but many mistakes still occurred in our data. For example, the attributes NDX, NPR, and NCHRONIC are capped by the AHRQ, but some hospitals still report greater values, which we simply ignored.

In extracting NPR, NDX, and NCHRONIC we also faced the second complication, namely that the number of values changed (increasing from 16 to 26) in 2009. One other attribute whose format changed is AGE. In 2012, for privacy reasons the AHRQ mandated that ages be "top-coded" (i.e. all values above a threshold be grouped into one category) at 90 in all samples. Prior HCUP data was not top-coded; however, for our experiments we chose to top-code all AGE data for two main reasons: (1) to ensure results for AGE are comparable across years and (2) to make our experiments address practical security risks to real deployments (in which ages may be top-coded). In Section 3 we discuss how this impacts the accuracy of our clique-finding attack.

The only attribute which did not appear in all years (namely not in 2004) of HCUP data was NCHRONIC. Since we had several other datasets for that attribute and the performance of all attacks on that attribute was similar across experiments, we were not concerned. We were not able to obtain the full 2008 NIS, and as a result we did not have MRISK or SEV data for that year. Our attacks performed well on the 2004 and 2009 MRISK and SEV attributes.

		# Patient records per hospital				
Year	**# Hospitals**	**Min**	**25%**	**50%**	**75%**	**Max**
2004	1004	15	1199	4300	11523	71580
2008	1056	3	889	3439	11170	117372
2009	1050	1	750	3278	10487	121668

Attribute name	**Abbrev.**	**Size**	**# Queries**	**2004**	**2008**	**2009**
Age (in days)	AGEDAY	365	66795	✓	✓	✓
Length of stay	LOS	365	66795	✓	✓	✓
Age (years)	AGE	91	4186	✓	✓	✓
Admission month	AMONTH	12	78	✓	✓	✓
# Chronic conditions	NCHRONIC	16	136		✓	
	NCHRONIC	26	351			✓
# Diagnoses	NDX	16	136	✓	✓	
	NDX	26	351			✓
# Procedures	NPR	16	136	✓	✓	
	NPR	26	351			✓
ZIP code income quartile	ZIPINC	4	10	✓	✓	✓
Mortality risk	MRISK	4	10	✓		✓
Disease severity	SEV	4	10	✓		✓

Figure 7: (Top) Number of hospitals and quartiles for number of records per hospital for 2004, 2008 and 2009 HCUP data. (Bottom) Attributes used in our experiments, their sizes, abbreviations, and their availability for each target year.